**Escuela de Ingeniería**

**Ingeniería Biomédica**

Ciclo escolar 2023 - 2024

Octavo semestre

Grupo A

**Robótica biomédica**

# Realidad Virtual: Avatar.

**Nombre del alumno:**

Hernández Ocón Luz Andrea


13 de mayo del 2024

## 1. INTRODUCCIÓN

Los avatares controlados por Unidades de Medición Inercial (IMU) representan un avance significativo en la realidad virtual. Estos avatares, también conocidos como avatares de cuerpo completo, son representaciones digitales de usuarios que pueden replicar con precisión los movimientos y gestos del cuerpo humano en entornos virtuales. La tecnología IMU permite el seguimiento en tiempo real de la orientación y posición del cuerpo a través de una combinación de acelerómetros, giroscopios y magnetómetros.

Los sensores IMU están integrados en un traje o dispositivo portátil que el usuario lleva consigo, permitiendo capturar los movimientos del cuerpo en tres dimensiones. Esta información se transmite a un sistema informático que procesa los datos y los aplica al avatar virtual, generando una representación en tiempo real de los movimientos del usuario en el mundo virtual.

La principal ventaja de los avatares controlados por IMU es su capacidad para ofrecer una experiencia de interacción natural y fluida en entornos virtuales. Los usuarios pueden moverse libremente y realizar gestos con el cuerpo de manera intuitiva, lo que aumenta la inmersión y la sensación de presencia en el mundo virtual. Esto los hace ideales para aplicaciones de realidad virtual, donde la interacción física es esencial para una experiencia envolvente.

Las aplicaciones de los avatares controlados por IMU son diversas y van desde el entretenimiento hasta la medicina y la industria. En el ámbito del entretenimiento, estos avatares se utilizan en videojuegos y experiencias de realidad virtual para permitir a los jugadores interactuar con el entorno y otros personajes de manera más realista. En la medicina, se emplean en la rehabilitación física para ayudar a los pacientes a recuperar la movilidad y la función motora.

## 2. OBJETIVO

Recrear un modelo virtual humanoide AVATAR incluyendo sensor de Aceleración y potenciómetros en articulaciones conectadas a un modelo real. El modelo virtual replica el modelo Real.

## 3. MARCO TEÓRICO

### 3.1 Unidad de Medición Inercial

IMU, que significa Unidad de Medición Inercial, es un dispositivo electrónico que mide e informa de la aceleración, la orientación, las velocidades angulares y otras fuerzas gravitatorias. Se compone de 3 acelerómetros, 3 giroscopios y 3 magnetómetros. Uno por eje para cada uno de los tres ejes del vehículo: yaw, pitch y roll.

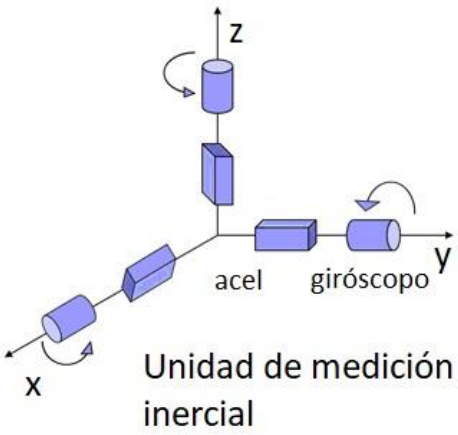


Imagen 1. Ejes del IMU

Dependiendo de la categoría del sensor IMU, las principales aplicaciones incluyen control y estabilización, navegación y corrección, o medición y pruebas. Sin embargo, los mercados típicos de las unidades de medición son el control de sistemas no tripulados, las aplicaciones cartográficas móviles ya sean terrestres, aéreas o marítimas y toda carga útil que requiera estabilización o apuntamiento.

### 3.2 Realidad virtual

La realidad virtual se podría definir como un sistema informático que genera en tiempo real representaciones de la realidad, que de hecho no son más que ilusiones

ya que se trata de una realidad perceptiva sin ningún soporte físico y que únicamente se da en el interior de los ordenadores.
La simulación que hace la realidad virtual se puede referir a escenas virtuales, creando un mundo virtual que sólo existe en el ordenador de lugares u objetos que existen en la realidad. También permite capturar la voluntad implícita del usuario en sus movimientos naturales proyectándolos en el mundo virtual que estamos generando, proyectando en el mundo virtual movimientos reales.

Las aplicaciones que en la actualidad encontramos de la realidad virtual a actividades de la vida cotidiana son muchas y diversas. Hay que destacar: la medicina, la simulación de multitudes y la sensación de presencia.

La aplicación en la medicina la encontramos en la simulación virtual del cuerpo humano. A partir de imágenes de nuestro cuerpo, se puede hacer la recreación en 3D del paciente, cosa que facilita la elaboración de un diagnóstico, o la simulación de operaciones en caso que sea necesario.

### 3.3 Avatar de realidad virtual

Un avatar con integración en realidad aumentada (AR), es una representación digital de un usuario o un personaje que logramos superponer en el mundo real a través de la tecnología de web AR. Se trata de una entidad virtual que interactúa con el entorno físico y el entorno del usuario en tiempo real. Los avatares se crean utilizando imágenes generadas por computación y están diseñados para integrarse en diferentes entornos, como plataformas de videojuegos, ecosistemas VR o el propio AR que lo incorpora a escenarios reales del usuario, mejorando la sensación de inmersión e interacción.

La tecnología web AR ha surgido como una de las tecnologías más transformadoras y útiles porque nos permite unir el entorno físico y el digital de forma más rápida y accesible.

En este sentido encontramos un punto de unión fascinante con el uso de avatares a través de AR, que abre nuevas posibilidades en diversos campos de aplicación

- **Garantizando accesibilidad al usuario común hacia nuevos formatos y contenidos digitales**

Los avatares sirven como un puente que conecta a los usuarios con los entornos virtuales. Cuando creamos estas representaciones digitales permitimos que los usuarios puedan interactuar de manera más natural con objetos y personajes digitalizados, fomentando una sensación de inmersión cercana y orgánica. Los avatares se pueden personalizar para que se parezcan a los propios usuarios, creando una experiencia más personal y amigable que permite conectar mejor con el usuario final.

- **Mejora de la Comunicación y la Interacción con el usuario**

En experiencias web AR podemos confirmar que los avatares desempeñan un papel crucial si queremos enriquecer la comunicación con los usuarios. Los avatares nos ofrecen múltiples posibilidades a la hora expresar emociones y señales no verbales o movimientos.

## 4. DESARROLLO

Se ha desarrollado un avatar virtual creado en VRealm Builder con articulaciones para facilitar su movimiento. El avatar cuenta con un sistema de padre-hijo, donde el torso es el padre de los brazos y las piernas.

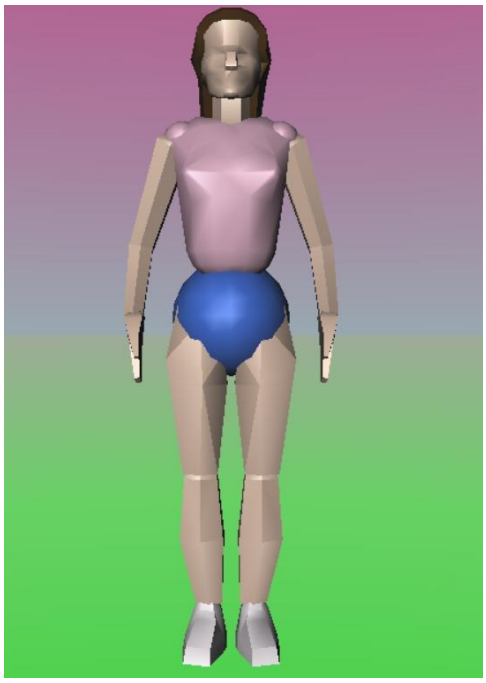

Imagen 2. Avatar de VRealm Builder

Este avatar es controlado a través de una muñeca articulada equipada con un Unidad de Medición Inercial (WT901C-TTL) colocada en el torso, y dos potenciómetros ubicados en los brazos. Para poder controlar los potenciómetros con los brazos, se colocaron perillas mini para potenciómetros en los hombros de la muñeca.

Imagen 3. IMU WT901C-TTL

Imagen 4. Muñeca con perillas para potenciómetros

El IMU registra la orientación y posición del torso, mientras que los potenciómetros capturan los ángulos de los brazos. Estos datos son recopilados por un código en Matlab, el cual procesa la información proveniente de los sensores para generar los movimientos correspondientes del avatar en el entorno virtual.

Imagen 5. Avatar funcionando con los movimientos de la muñeca

A través del código de Matlab, es posible visualizar en tiempo real los movimientos del avatar, lo que facilita el análisis y la evaluación de su comportamiento.

Finalmente, se realizó un diseño de caja para la estructura de la muñeca y los potenciómetros en FUSION 360.

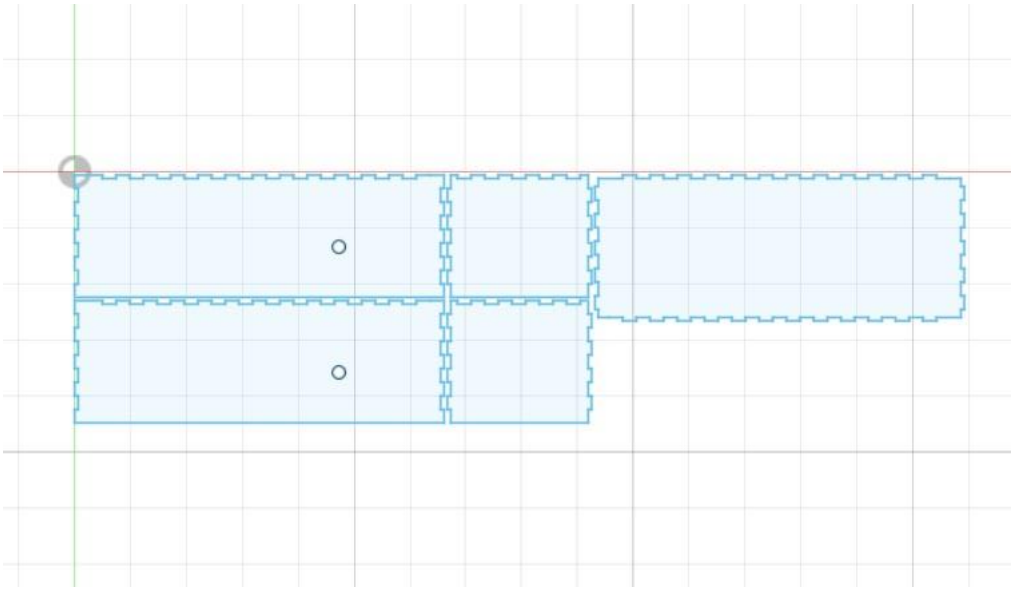

Imagen 6. Estructura para la muñeca y potenciómetros

## 5. RESULTADOS

Se obtuvo un avatar funcional que realiza los movimientos de flexión y extensión de la cadera. Igualmente, la flexión y extensión de ambos brazos siguiendo los movimientos realizados por la muñeca.

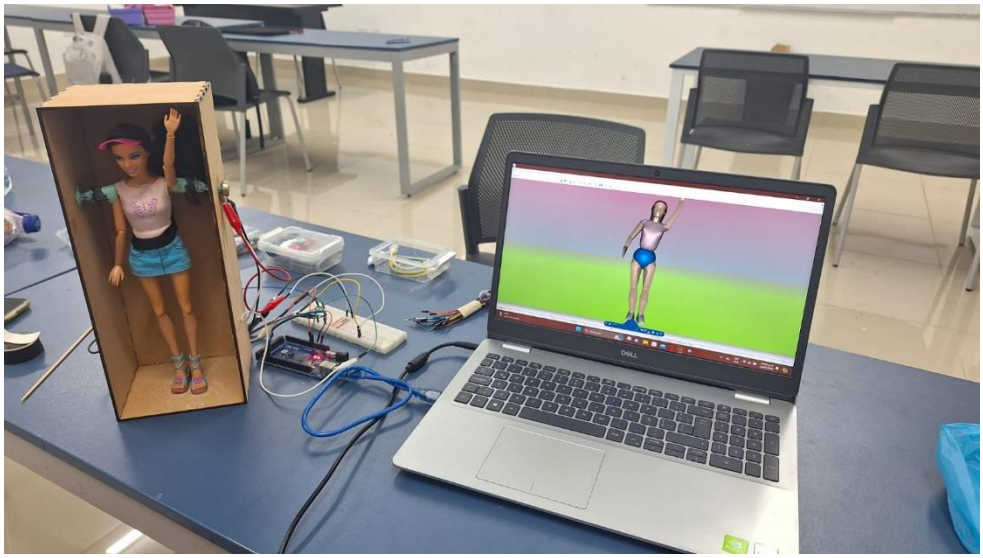

Imagen 7. Avatar en funcionamiento

Finalmente, se obtuvo el código en Matlab, el cual se encarga de obtener los ángulos de los potenciómetros y de la IMU.

```
#include "REG.h"
#include "wit_c_sdk.h"
const int pots[] = {A0, A1};
/*
Test on MEGA 2560. use WT901CTTL sensor

WT901CTTL     MEGA 2560
   VCC <--->  5V/3.3V
   TX  <--->  19(TX1)
   RX  <--->  18(RX1)
   GND <--->  GND
*/

#define ACC_UPDATE                  0x01
#define GYRO_UPDATE                 0x02
#define ANGLE_UPDATE        0x04
#define MAG_UPDATE                  0x08
#define READ_UPDATE                 0x80
static volatile char s_cDataUpdate = 0, s_cCmd = 0xff;

static void CmdProcess(void);
static void AutoScanSensor(void);
static void SensorUartSend(uint8_t *p_data, uint32_t uiSize);
static void SensorDataUpdata(uint32_t uiReg, uint32_t uiRegNum);
static void Delayms(uint16_t ucMs);
const uint32_t c_uiBaud[8] = {0,4800, 9600, 19200, 38400, 57600, 115200, 230400};
float giros[2] = {0};
float grados[2] = {0};
```

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
        WitInit(WIT_PROTOCOL_NORMAL, 0x50);
        WitSerialWriteRegister(SensorUartSend);
        WitRegisterCallBack(SensorDataUpdata);
  WitDelayMsRegister(Delayms);
        //Serial.print("\r\n******** wit-motion normal example  ********\r\n");
        AutoScanSensor();
}
int i;
float fAcc[3], fGyro[3], fAngle[3];
void loop() {
    while (Serial1.available())
    {
      WitSerialDataIn(Serial1.read());
    }
    while (Serial.available())
    {
      CopeCmdData(Serial.read());
    }
                CmdProcess();
                if(s_cDataUpdate)
                {
                        for(i = 0; i < 3; i++)
                        {
                                fAcc[i] = sReg[AX+i] / 32768.0f * 16.0f;
                                fGyro[i] = sReg[GX+i] / 32768.0f * 2000.0f;
                                fAngle[i] = sReg[Roll+i] / 32768.0f * 180.0f;
                        }
                        if(s_cDataUpdate & ANGLE_UPDATE)
                        {

                                Serial.print(fAngle[0]);
       Serial.print(',');
                                Serial.print(fAngle[1]);
       Serial.print(',');
                                //Serial.print(fAngle[2]);
     //  Serial.print(',');
       delay(100);
                                s_cDataUpdate &= ~ANGLE_UPDATE;


                        }
     s_cDataUpdate = 0;
                }

    for (int i = 0; i < 2; i++) {
    giros[i] = analogRead(pots[i]) * 100.0 / 1023.0;
    grados[i] = map(giros[i], 0, 100, 0, 290);
  }

  Serial.print(grados[0]);
  Serial.print(',');
  Serial.print(grados[1]);
  Serial.println();
  delay(100);
}


void CopeCmdData(unsigned char ucData)
{
        static unsigned char s_ucData[50], s_ucRxCnt = 0;

        s_ucData[s_ucRxCnt++] = ucData;
        if(s_ucRxCnt<3)return;
              //Less than three data returned
        if(s_ucRxCnt >= 50) s_ucRxCnt = 0;
        if(s_ucRxCnt >= 3)
```

```
        {
            if((s_ucData[1] == '\r') && (s_ucData[2] == '\n'))
            {
                    s_cCmd = s_ucData[0];
                    memset(s_ucData,0,50);
                    s_ucRxCnt = 0;
            }
            else
            {
                    s_ucData[0] = s_ucData[1];
                    s_ucData[1] = s_ucData[2];
                    s_ucRxCnt = 2;

            }
        }
}
static void CmdProcess(void)
{
        switch(s_cCmd)
        {
            case 'a':   if(WitStartAccCali() != WIT_HAL_OK) Serial.print("\r\nSet AccCali Error\r\n");
                        break;
            case 'm':   if(WitStartMagCali() != WIT_HAL_OK) Serial.print("\r\nSet MagCali Error\r\n");
                        break;
            case 'e':   if(WitStopMagCali() != WIT_HAL_OK) Serial.print("\r\nSet MagCali Error\r\n");
                        break;
            case 'u':   if(WitSetBandwidth(BANDWIDTH_5HZ) != WIT_HAL_OK) Serial.print("\r\nSet Bandwidth Error\r\n");
                        break;
            case 'U':   if(WitSetBandwidth(BANDWIDTH_256HZ) != WIT_HAL_OK) Serial.print("\r\nSet Bandwidth Error\r\n");
                        break;
            case 'B':   if(WitSetUartBaud(WIT_BAUD_115200) != WIT_HAL_OK) Serial.print("\r\nSet Baud Error\r\n");
          else
          {
           Serial1.begin(c_uiBaud[WIT_BAUD_115200]);
           //Serial.print(" 115200 Baud rate modified successfully\r\n");
          }
                        break;
            case 'b':   if(WitSetUartBaud(WIT_BAUD_9600) != WIT_HAL_OK) Serial.print("\r\nSet Baud Error\r\n");
          else
          {
           Serial1.begin(c_uiBaud[WIT_BAUD_9600]);
           Serial.print(" 9600 Baud rate modified successfully\r\n");
          }
                        break;
            case 'r': if(WitSetOutputRate(RRATE_1HZ) != WIT_HAL_OK)  Serial.print("\r\nSet Baud Error\r\n");
                            else //Serial.print("\r\nSet Baud Success\r\n");
                        break;
            case 'R':   if(WitSetOutputRate(RRATE_10HZ) != WIT_HAL_OK) Serial.print("\r\nSet Baud Error\r\n");
          else Serial.print("\r\nSet Baud Success\r\n");
                        break;
   case  'C':  if(WitSetContent(RSW_ACC|RSW_GYRO|RSW_ANGLE|RSW_MAG)  !=  WIT_HAL_OK)  Serial.print("\r\nSet  RSW
        Error\r\n");
     break;
   case 'c': if(WitSetContent(RSW_ACC) != WIT_HAL_OK) Serial.print("\r\nSet RSW Error\r\n");
     break;
            case 'h':   ShowHelp();
                        break;
            default :break;
        }
        s_cCmd = 0xff;
}
static void SensorUartSend(uint8_t *p_data, uint32_t uiSize)
{
  Serial1.write(p_data, uiSize);
  Serial1.flush();
}
static void Delayms(uint16_t ucMs)
{
  delay(ucMs);
}
```

```
static void SensorDataUpdata(uint32_t uiReg, uint32_t uiRegNum)
{
        int i;
   for(i = 0; i < uiRegNum; i++)
   {
      switch(uiReg)
      {
         case AZ:
                                        s_cDataUpdate |= ACC_UPDATE;
         break;
         case GZ:
                                        s_cDataUpdate |= GYRO_UPDATE;
         break;
         case HZ:
                                        s_cDataUpdate |= MAG_UPDATE;
         break;
         case Yaw:
                                        s_cDataUpdate |= ANGLE_UPDATE;
         break;
         default:
                                        s_cDataUpdate |= READ_UPDATE;
                        break;
      }
            uiReg++;
   }
}

static void AutoScanSensor(void)
{
        int i, iRetry;

        for(i = 0; i < sizeof(c_uiBaud)/sizeof(c_uiBaud[0]); i++)
        {
            Serial1.begin(c_uiBaud[i]);
   Serial1.flush();
            iRetry = 2;
            s_cDataUpdate = 0;
            do
            {
                        WitReadReg(AX, 3);
                        delay(200);
    while (Serial1.available())
    {
     WitSerialDataIn(Serial1.read());
    }
                        if(s_cDataUpdate != 0)
                        {
                                    //Serial.print(c_uiBaud[i]);
                                    //Serial.print(" baud find sensor\r\n\r\n");
                                    ShowHelp();
                                    return ;
                        }
                        iRetry--;
            }while(iRetry);
        }
        Serial.print("can not find sensor\r\n");
        Serial.print("please check your connection\r\n");
}
```

## Y el código de Arduino que se encarga de la conexión serial y de la obtención de los datos

```
clc
if ~isempty(instrfind)
   fclose(instrfind);
   delete(instrfind);
end

numDatos = 1000;
```

```
PuertoSerial = 'COM9';

% Access the 3D World from MATLAB
world = vrworld('Monita.wrl', 'new');
open(world);
fig = vrfigure(world);

% Access the nodes in the avatar
%set(fig, 'Viewpoint', 'Far View');
r1 = vrnode(world, 'X');
r2 = vrnode(world, 'Y');
r3 = vrnode(world, 'Hombro_derecho');
r4 = vrnode(world, 'Hombro_izquierdo');

i = 1;

s = serial(PuertoSerial);
set(s, 'BaudRate', 115200);
fopen(s);
disp('Presiona enter para continuar');
pause();                        % pausa y espera a que el usuario presione cualquier tecla

while (i < numDatos)
   datos = fscanf(s, '%f,%f,%f,%f\n');   % espera una cadena de caracteres que comience con el carácter '$' y tenga 3 datos float

   angulo1 = datos(1);             % ángulos recibidos desde Arduino para el primer node
   angulo2 = datos(2);
   angulo3 = datos(3);             % ángulos recibidos desde Arduino para el primer node
   angulo4 = datos(4);
   angulor1 = ((angulo1*2*pi)/180)             % ángulos recibidos desde Arduino para el primer node
   angulor2 = ((angulo2*2*pi)/180)
   angulor3 = ((angulo3*2*pi)/180)             % ángulos recibidos desde Arduino para el primer node
   angulor4 = ((angulo4*2*pi)/180)
   % Aplica rotación a los nodos r1, r2, r3

   if angulor1>0.6
     angulor1 = 0.6;
      elseif angulor1<-0.6
     angulor1 = -0.6;
      elseif angulor2>1
      angulor2 = 1;
      elseif angulor2<-1
      angulor2 = -1;
      else
      angulor1=angulor1;
      angulor2=angulor2;
   end

  r1.rotation = [0 0 1 angulor1];
  r2.rotation = [1 0 0 angulor2];
  r3.rotation = [1 0 0 angulor3];
  r4.rotation = [1 0 0 angulor4];

   % Update the figure
   vrdrawnow;

   i = i + 1;  % aumenta el bucle

end

fclose(s);  % cierra la conexion
delete(s);
clear s;
```

## 6. CONCLUSIONES

En conclusión, la práctica demuestra la posibilidad de utilizar tecnologías de sensores como IMU y potenciómetros para controlar avatares virtuales en entornos como VRealm Builder. La integración de estos dispositivos permite una mejor interacción con el avatar, ya que los movimientos del usuario se reflejan con precisión en el mundo virtual.

Además, el uso de Matlab como softaware de visualización proporciona una herramienta fácil de usar para procesar los datos de los sensores y visualizar los movimientos del avatar en tiempo real.

## 7. REFERENCIAS

SBG Systems. (2023). IMU - Unidad de medición inercial. Recuperado de: https://www.sbg-systems.com/es/unidad-de-medicion-inercial-sensor-imu/#:~:text=IMU%2C%20que%20significa%20Unidad%20de,necesidades%20de%20cabo%20%2C%203%20magnet%C3%B3metros.

Facultat d'Informàtica de Barcelona (s.f.) Realidad virtual. Recuperado de: https://www.fib.upc.edu/retro-informatica/avui/realitatvirtual.html

Onirix (2023) Avatares en realidad aumentada, que aplicaciones tiene y cómo podemos crearlos. Recuperado de: https://www.onirix.com/es/avatares-en-realidad-aumentada-que-aplicaciones-tiene-y-como-podemos-crearlos/